AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 1, line 14, as follows:

Briefly, 'SSA form' is an alternative representation for variables in a program, in which any given variable is only assigned at a ~~singe~~ single location in the program. A program is transformed into SSA form by a process called 'SSA conversion'. The SSA conversion replaces every local variable in the source program with a set of new variables, called 'SSA variables', each of which is only assigned to at a single physical location in the program; thus, every point at which a source variable V is assigned to in the source program, the corresponding SSA-converted program will instead assign a unique variable, V' l, V'2, etc.

Please amend the paragraph bridging pages 1 and 2, beginning at page 1, line 22, as follows:

At any point in the program (always at the start of a basic-block) where the merging of control flow would cause two such derived variables to be live ~~simultaneously~~ simultaneously, their values are merged together to ~~yielding~~ yield a single new SSA variable, e.g., V'3, that represents the value of the original source variable at that point. This merging is done using a 'phi-function'. The 'phi-function' is an instruction which has as many inputs as there are basic-blocks that can transfer control to the basic-block it is in, and chooses whichever input corresponds to the basic-block that preceded the current one in the dynamic control flow of the program.

Please amend the subparagraph beginning at page 3, line 14, as follows:

(b) (202) Every ~~non-SA~~ non-SSA variable definition is replaced by a definition of a unique SSA-variable, and every non-SSA variable reference replaced by a reference to an

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

appropriate SSA-variable[[,]]. ~~because~~ Because of the insertion of phi-functions, there will always be a single extant SSA-variable corresponding to a given non-SSA variable.

Please amend the paragraph beginning at page 3, line 22, as follows:

The concept of basic-block 'dominance' is well known, and can be described as follows: A basic block A 'dominates' a basic ~~black~~ block B, if the flow of control can reach B only after A (although perhaps not immediately; other basics blocks may be executed between them).

Please amend the paragraph beginning at page 4, line 16, as follows:

In this invention, we modify the procedure of ~~[SSAFFORM]~~ [SSAFORM], which is shown in Figure 12, as follows:

Method for representing pointer variables in SSA form ~~in step (452)~~

+ References or definitions of memory locations resulting from pointer-dereferences are also treated as 'variables', here called 'complex variables' ~~shown in (452)~~, in addition to simple variables[[(451)]], such as those used in the source program. Complex variables consist of a pointer variable and an offset from the pointer. An example of a complex variable is the C source expression (value) '*P', as used in (810) and (820).

Please amend the paragraph bridging pages 4 and 5, beginning at page 4, line 25, as follows:

Method for adding appropriate copy operations to synchronize complex variables ~~(452)~~ with the memory location they represent in Figure 1. +These 'complex ~~variables~~' variables' ~~(452)~~ are treated as non-SSA variables during SSA-conversion shown in Figure 1 (any variable reference within a complex variable is treated as a reference in the instruction (440) that contains the complex variable ~~(452)~~).

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

Please amend the paragraph beginning at page 5, line 5, as follows:

+A new step (120) is inserted in the SSA-conversion process as shown in Figure l between steps (a) (110) and (b) (130), to take care of any necessary synchronization of SSA-converted complex variables ~~(452)~~ with any instructions (440) that have unknown side-effects:

Please amend the subparagraph beginning at page 5, line 9, as follows:

(a') (121) To any instruction (440) that may have unknown side-effects on an 'active' complex variable ~~(452)~~ -- one that is defined by some dominator of the instruction -- add a list of the variable, and the possible side effects (may_read, may_write).

Please amend the paragraph beginning at page 5, line 13, as follows:

(122, 123) Next, insert special copy operations, called write-backs (521) (which write an SSA variable back to its real location) and read-backs (which define a new SSA variable from a variable's real location), to make sure the SSA-converted versions of affected variables (450) correctly synchronized with respect to such side-effects. This step may also insert new phi-functions, in the case where copying back a complex variable ~~(452)~~ from ~~it's synchronizatian~~ its synchronization location may define a new SSA version of that variable.

Please amend the paragraph beginning at page 6, line 20, as follows:

Figure 12 shows general ~~farm~~ form of the traditional SSA-conversion process.

Please amend the paragraph bridging pages 7 and 8, beginning at page 7, line 24, as follows:

An instruction (440) may be a function call, in which case it can have arbitrary side-effects, but control-flow must eventually return to the instruction (440) following the function ~~call~~ call.

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

Please amend the paragraph beginning at page 8, line 2, as follows:

An instruction (440) may explicitly read or write 'variables' (450), each of which is either a 'simple variable' ~~(451)~~, such as a local or global variable in the source program (or a temporary variable created by the compiler), or a 'complex variable' ~~(452)~~, which represents a memory location that is indirectly referenced through another variable. Each variable has a type, which defines what values may be stored in the variable.

Please amend the paragraph beginning at page 8, line 7, as follows:

Complex variables ~~(452)~~ are of the form '*(BASE + OFFSET)', where BASE ~~(453)~~ is a variable (450), and OFFSET ~~(454)~~ is a constant offset; this notation represents the value stored at memory location (BASE + OFFSET).

Please amend the paragraph beginning at page 8, line 10, as follows:

Because of the use of complex variables ~~(452)~~, there are typically no instructions (440) that serve to store or retrieve values from a computed memory location. Instead, a simple copy where either the source or destination, or both, is a complex variable ~~(452)~~ is used. Similarly, any other instruction (440) may store or retrieve its results and operands from memory using complex variables.

Please amend the paragraph beginning at page 9, line 3, as follows:

(a') I. (121) For each operation, determine which 'active' complex variables ~~(452)~~ it may have unknown side-effects on, and list attach a note to the operation with this information. These notes are referred to below as 'variable syncs'. In the example program, instructions (1020), (1021), (1022), and (1023) may possibly read or modify '*P', (as we don't have any information about them).

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

Please amend the paragraph beginning at page 9, line 8, as follows:

II. (122) At the same time, add any necessary write-back copy operations (521) write back any complex variables ~~(452)~~ to their 'synchronization ~~location'~~ location', which is the original non-SSA variable (which, for complex variables ~~(452)~~, is a memory location), and mark the destination or the copy operation as such (this prevents step (b) of SSA conversion from treating the destination of the copy as a new SSA definition). Any such 'write-back' (521) makes the associated variable inactive, and so prevents any further write-backs (521) unless the variable is once again defined.

Please amend the paragraph beginning at page 9, line 15, as follows:

III. (123) Add necessary read-backs, to supply new SSA definitions of complex variables ~~(452)~~ that have been invalidated (after having been written back to their synchronization location).

Please amend the subparagraph beginning at page 9, line 20, as follows:

+ Defined by operations that may modify a complex variable ~~(452)~~, as located in step 1 above, or by the merging of multiple active read-backs of the same variable ~~(450]~~ (450), at control-now merge points. In the example, all the function ~~call~~ calls may possibly modify '*P', so they must be represented by read-backs at (1020), (1021), ~~(1022)~~ (1022), and (1024).

Please amend the paragraph beginning at page 10, line 8, as follows:

+ Killed by definitions of the associated complex variable ~~(452)~~, or by a new read-back of the variable. In the example, the read-back defined at (1021) is killed because the following function call defines a new read-back of the same variable at (1022).

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

Please amend the paragraph beginning at page 10, line 16, as follows:

After a fixed-point of read-back definitions is reached, those that are referenced are instantiated by inserting the appropriate copy operation at the place where they are defined, to copy the value from the read-back variable (450)'s synchronization location into a new SSA variable; if necessary new phi-functions may be inserted to reflect this new definition ~~pint~~ point. As mentioned above, in the example this only happens at ~~(1030)~~ (1030).

Please amend the paragraph beginning at page 12, line 1, as follows:

(702) ~~Initiaiize~~ Initialize the queue PENDING_BLOCKS to the function's entry block.

Please amend the paragraph beginning at page 12, line 4, as follows:

(720) For each read-back RB in any block (430) that has been marked as 'used', and (721) isn't a 'merge read-back' ~~who's~~ whose sources (the read-backs that it merges) are all also marked 'used', instantiate that read-back as follows:

(730) If RB is a 'merge read-back', then the point of read-back is (741) the beginning of the block (430) where the merge occurs, otherwise it is (742) immediately after the instruction (440) that created the read-back.

Please amend the paragraph bridging pages 12 and 13, beginning at page 12, line 23, as follows:

(810) Calculate the intersection of the end read-back sets for each predecessor block (431) of BLOCK in the flow-graph, calling the result NEW_BEGIN_READ_BACKS. The intersection is calculated as ~~follows~~ follows:

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567

Any predecessor read-back for which a read-back of the same variable doesn't exist in one of the other predecessor blocks is discarded from the result; it is also marked as 'referenced'.

Please amend the paragraph beginning at page 13, line 4, as follows:

If the read-back for a given ~~viable~~ variable is the same read-back in all predecessor blocks (431), that read-back is added to the result.

Please amend the paragraph bridging pages 13 and 14, beginning at page 13, line 23, as follows:

(860) For each variable sync in INSTRUCTION that notes a variable ~~VAR1ABLE~~ VARIABLE as possibly written, do:

(865) Add a new read-back entry for VARIABLE to NEW_END_READ_BACKS, replacing any existing read-back of VARIABLE.

Please amend the paragraph beginning at page 14, line 10, as follows:

The exception to this rule is complex variables ~~(452)~~ that have been marked as special 'synchronization' locations, in the copy instruction (440) inserted in step (a'); they are left as-is, referring to the original complex variable ~~(452)~~.

HAYES SOLOWAY P.C.
130 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567